A QUICK UPLINK, EXPANDABLE EXECUTABLE FOR THE NSSC-I OF THE HUBBLE SPACE TELESCOPE

Glenn T. Foley and Elizabeth A. Citrin

NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

The Hubble Space Telescope, launched in April 1990, contains two primary on-board computers. In the past, modifications to the flight software have been accomplished via patches to the on-board executable image (performed while software is executing), or via halt and reload of the computer memory with a new flight software version. This paper describes a method of reloading flight software with a new software version while continuing the on-board execution of a reduced set of spacecraft payload functions.

Key Words: Aerospace, operations, flight software, maintenance

1. Introduction

The Hubble Space Telescope (HST), launched in April 1990, is a 2.4 meter aperture telescope system designed to provide observational capabilities well beyond that of existing ground-based telescopes. This unique orbiting facility is supported by a combination of dedicated and institutionally-provided flight and ground systems which enable science planning and scheduling, mission and science control operations, and data acquisition, processing, analyses, and archival. The HST is planned to operate for at least 15 years.

The HST contains two primary on-board computers. Both computers, designed in the 1970's, are severely memory constrained in relation to current HST flight software requirements, and post-launch experience has necessitated frequent, relatively extensive software modifications. This paper examines the various methods available to install software modifications and discusses, specifically, software modifications relating to one of the HST on-board computers, the NSSC-I (NASA Standard Spacecraft Computer, Model I).

1.1 HST Observatory

The HST Observatory, the on-orbit portion of the HST, has three major components: The optical telescope assembly (OTA), the Support Systems Module (SSM) and the payload. Figure 1 shows the relationship among these subsystems.

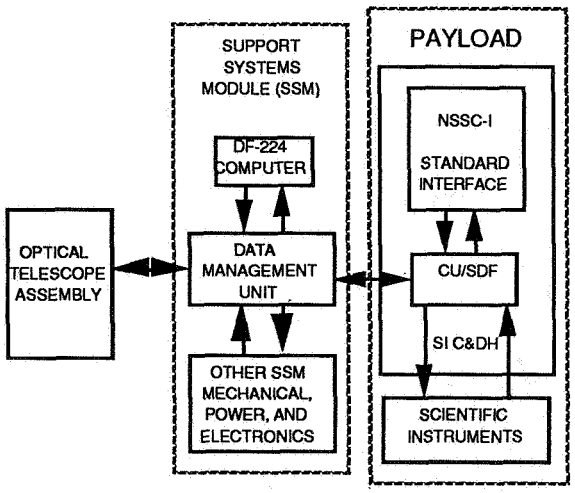

Figure 1. HST Major Subsystems

The OTA is a 2.4-meter Ritchey-Chretian telescope consisting of a primary and a secondary mirror. The telescope's focal plane is divided among four axial science instruments (SIs), one radial SI, and three Fine Guidance Sensors (FGSs). The FGSs perform both pointing control and astrometry functions. The OTA also includes thermal and optics control electronics.

The SSM contains the data management system, including the central computer for the

spacecraft, the DF-224. The SSM provides thermal control, electrical power, communications, data management, and pointing control in support of the OTA, OTA equipment section, the payload, and other SSM subsystems. The SSM data management system interfaces to the payload's Scientific Instrument Control and Data Handling (SI C&DH) module.

In addition to the SI C&DH, the payload has a complement of five scientific instruments that includes two cameras, two spectrographs, and a photometer. Within the SI C&DH, the Control Unit/Science Data Formatter (CU/SDF) and the NSSC-I manage commands and telemetry within the payload.

1.2 NSSC-I Computer

The NSSC-I computer includes a fixed-point arithmetic central processing module and 64K words of random access memory. HST flight software residing in the NSSC-I includes two basic components, the flight executive (Executive) and SI application software. All of the NSSC-I flight software is written in assembly language.

The Executive software is described in detail in the *Multimission Modular Spacecraft On-Board Computer Flight Executive Technical Description* (Ref. 1). This software is responsible for processing commands to the HST payload (initiated from the ground, other NSSC-I software, or stored command memory), and collecting telemetry from the payload and forwarding it to the SSM for transmission to the ground. The Executive performs safety monitoring functions for the payload. The Executive is also responsible for scheduling the execution of other software processors within the NSSC-I, and for performing diagnostic self tests to ensure integrity of the NSSC-I and its software.

The SI application software in the NSSC-I is tailored to the requirements of each science instrument. Generally speaking, this software performs functions such as configuring the SI mechanisms and optics for use, acquisition of astronomical targets, observations with the instrument, health and safety monitoring, and processing and quality checking of science data.

2. Changing the NSSC-I Software

Throughout the early lifetime of the HST mission, desired enhancements to the functionality of the payload control computer have been continually identified, from improved on-board target acquisition algorithms for the science instruments, to increased diagnostic information for daily operations. In addition, as some of the payload hardware has aged, failures and reduced sensitivities have warranted correction in the form of software fixes and workarounds. Some of these more critical changes were implemented as emergency software updates, but most were simply added to a list of desirable new capabilities. As HST settled into normal operations after about a year in orbit, an effort was begun to implement about forty such new desired capabilities. This new product was named the Baseline 4.0 software for the NSSC-I.

When new requirements for the NSSC-I software have been identified, it is necessary to define a method for installing the changes on-board the spacecraft. Ideally, we wish to minimize risk to the health of the spacecraft as well as the dedicated observatory time needed to make the changes become part of the operational system. Several software installation options are available. Selection of a method depends on the quantity of changes being implemented, how the design changes are distributed throughout the system, and how critical the NSSC-I function that they affect is.

2.1 Patching NSSC-I Memory

Because of the extreme cost associated with operating the observatory, it is desirable to install NSSC-I software changes without having to halt the computer and interrupt the timeline of scientific observations. Also, since spacecraft communication contacts are costly, it is beneficial to minimize the actual number of NSSC-I words that must be modified to implement a software change. For example, if a single piece of logic in a program is the only area affected by a change, it is desirable to overlay the new machine language on top of the old code (and spare memory, if necessary),

rather than reloading the entire affected program. This method is referred to as *patching*. Patching can provide an extraordinary savings in uplink time, but often requires very meticulous, bit-level composition of machine language instructions by hand. Needless to say, creating a patch can be a very time consuming and error prone process. In addition, long term maintenance may become an issue as the number of patches that have been performed on a baselined NSSC-I load module grow. Maintaining an accurate map of the pieces of available spare memory is a necessity and reclaiming fragmented space for desired new uses is not always possible. In addition to living with "spaghetti code," the patch versions of changes are inevitably less efficient than inline implementation of logic because of the extra memory words and CPU resources required to transfer control between all of the patch fragments. Some changes, by their very nature, are not patchable because they affect a critical realtime function - execution of that function when the patch is only partly installed could give disastrous results! In any case, the sheer magnitude and complexity of the NSSC-I Baseline 4.0 changes rendered implementation of the changes in patch format to not be a viable option.

2.2 A Hardware Reload of NSSC-I Memory

Given that uplinking Baseline 4.0 in patch format was not feasible, reloading all of NSSC-I memory was the only option available. The standard method for performing this is a *hardware memory load*. In a hardware load, the NSSC-I is halted, and each of the sixteen, 4096-word banks of memory is in turn loaded with its new contents. Though it avoids the difficulties involved in patching, a complete reload provides its own drawbacks. The greatest detractor is that for the period of reloading, the NSSC-I computer must be halted. This is a great impact to spacecraft operations because the science instruments must be commanded to a safe state and the science timeline must be stopped for the duration of the memory load. Also, detailed engineering telemetry describing the health of the payload is not available when the NSSC-I is not running - only a reduced set of fixed telemetry parameters are reported, providing spacecraft operators with reduced visibility into the state of the payload. Several orbits of spacecraft time are sacrificed for the numerous dedicated contacts required to perform the load. The hardware load process is also not very forgiving: if a single word in the middle of a bank load is dropped due to a communications dropout, the entire 4096-word bank must be reloaded from scratch.

Degradation of some elements in the HST payload has made hardware reloading of the NSSC-I even less desirable. Since the NSSC-I is responsible for the safety of the five Hubble science instruments, the SIs must be pre-commanded to a safe state before the NSSC-I is halted. Unfortunately, two of the SIs have experienced problems which make it desirable to keep them out of their respective safe states.

The Goddard High Resolution Spectograph (GHRS) suffers from an intermittent power supply failure that prevents it from transmitting its science data, effectively halting its observation. It was discovered that this problem is mitigated by thermally stabilizing the affected area by keeping one of the instrument's main electronics boxes (MEBs) powered on continuously. Unfortunately, this MEB cannot stay on without being monitored by the NSSC-I, and instrument engineers fear that thermal cycling of the failed area may eventually lead to a permanent failure of the faulty hardware. It would therefore be desirable to somehow allow the GHRS MEB to stay powered on for the period of the NSSC-I reload.

Similarly, the Wide Field and Planetary Camera (WFPC) instrument suffers from a contaminant in its optics. After the WFPC's thermo-electric coolers (TECs) drive the instrument's charge-coupled device (CCD) detectors to their cold operating temperature, a costly and time consuming decontamination and recalibration procedure is performed to drive the contaminant from the optics and compensate for its effects. Unfortunately, WFPC's safe state includes having its TECs powered off, which results in the instrument warming up and the contaminants vaporizing, only to recondense on the sensitive WFPC optics when the TECS are later turned back on. This requires the decontamination and recalibration procedure to again be performed. Therefore, it

would be desirable to allow the WFPC TECs to remain on during the NSSC-I reload.

2.3 A Software Reload of NSSC-I Memory

All NSSC-I memory loads that are performed when the NSSC-I is running, whether they are loads of stored command memory, SI observation control tables, or even the previously described software patches, are performed via *software memory load*. In a software load, realtime commands from the ground send up blocks of from one to 62 contiguous NSSC-I memory words and a destination where the words are to be loaded. The block is also tagged with a checksum word that allows retransmission of the block from the ground if the NSSC-I detects an error or data dropout. The unfortunate drawback of software loading is that since the NSSC-I must be running, all of memory could not be loaded using this method because one would overwrite the software performing the memory load with the software being loaded! Clearly, software memory loading on its own will not solve our problem.

3. Elements of the Ideal Reload

Thus far in our examination of patches and hardware and software memory loads, we have seen several elements that would be desirable during performance of our NSSC-I reload. We would like reload all of memory and not have to patch the new functions in, yet we would also like to have the NSSC-I running to provide nominal engineering telemetry visibility into the payload, some level of SI safing protection, and the ability to load memory in small "chunks," with some success/failure feedback to the ground system. In essence, we would like to have a subset of the existing NSSC-I Executive functionality.

Given that we could extract a core of the existing NSSC-I software, supplemented with minimal custom software, the next question would be how we could put this to use in the NSSC-I hardware environment. To address this, let us more closely examine the layout of NSSC-I memory. As discussed earlier, the 65,536 words of NSSC-I memory are divided into sixteen 4096-word banks. The first six of these banks contain program data, the next five and a half contain the program code, and the remaining four and a half are used for the timeline of stored commands that instruct the payload through its daily operations. In any of the NSSC-I reload approaches considered, standard payload operations would have to be temporarily suspended, even if only for a very short time. Given this inevitability, the stored command memory would not be required through this period. This gave rise to utilization of this memory to house the target reduced set of NSSC-I capabilities required to perform our reload. The software functions desired, together with the operational procedures and contingencies to make use of this software, were labelled the Quick Uplink, Expandable Executable for the NSSC-I, or QUEEN.

3.1 QUEEN Technical Description

A major advantage of the QUEEN approach is that it is composed of a subset of existing software. We shall now examine the QUEEN requirements, first in terms of capabilities of the full NSSC-I Executive, then by describing the software custom written for QUEEN. The design goal for memory utilization was to have QUEEN require less than two NSSC-I banks to operate: 4096 words for code, and 4096 words for program data.

To allow spacecraft operators full visibility into the health of the payload, QUEEN provides for collection of all normal mode engineering telemetry from the various payload elements, including insertion of special QUEEN operational status information.

NSSC-I memory loading and dumping will be performed by inclusion of two of the existing "Executive requests" to software memory load and dump NSSC-I memory. This is the core of the QUEEN concept. Any of the other Executive requests, such as "execute a program" or "dump a data log," are unceremoniously ignored.

QUEEN supports nominal intercomputer communications between the NSSC-I and the DF-224 flight computer through exchange of their processor interface tables (PITs) every half second. QUEEN maintains the NSSC-I clock synchronously with that reported by the

DF-224 in its PIT, and will continue to issue time code update commands to the payload control unit once per minute, as expected.

The QUEEN payload safing capabilities were derived as special cases from the more general safing protection offered by the full NSSC-I Executive. QUEEN safing consists of issuing commands to turn off critical subsystems in either the WFPC or GHRS, or for the entire NSSC-I/WFPC/GHRS complement, in the event of a systemic problem.

Minor modifications were made to the inherited portion of the Executive to allow access to the QUEEN-specific payload safing capabilities. QUEEN payload safing may be invoked by a request from the ground (in the form of a single word memory load to a predefined location), when commanded by the DF-224's PIT, or upon absence of a predefined number of DF-224 PITs. Loss of the "master timing pulse" or one megahertz clock control signals from the payload control unit will also cause QUEEN payload safing. For additional protection, the NSSC-I would signal the DF-224 through the PIT upon detection of a "delayed command error" from the payload control unit. Such an error would indicate a hardware failure that would prevent the NSSC-I from providing proper protection to the payload elements. In this case the DF-224 would command the payload to its safe state until the problem can be resolved.

Finally, QUEEN includes custom software that mimics the more general purpose engineering data limit checking capabilities of the full Executive. This allows on-board inspection of critical payload parameters against specified limits, resulting in safing of a payload element if an item remains out of limits for too long.

3.2 QUEEN Operational Usage

Actual implementation of the QUEEN NSSC-I code was only part of this approach to loading the flight software. Development of procedures to install and utilize QUEEN and prepare for any contingencies was also necessary. The list of contingencies, failure scenarios, and backout procedures is quite extensive - we shall discuss only the nominal installation procedure. Figure 2 presents the four main steps of QUEEN usage.

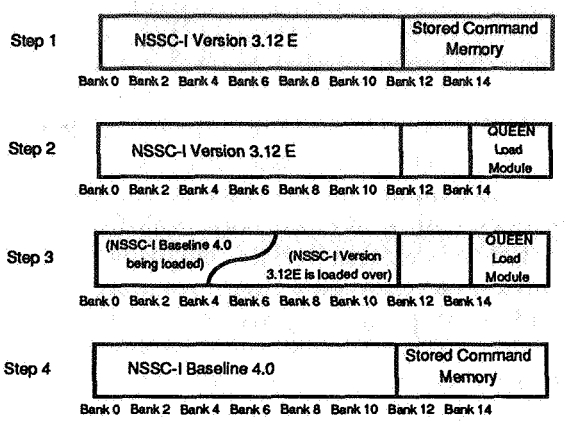

Figure 2. QUEEN Usage

First, with the previous version of the flight software running (Version 3.12E), the stored command timeline is stopped and all SIs except GHRS and WFPC are commanded to their safe states (Step 1). Next, under version 3.12E control, QUEEN is software loaded into the last 2 banks of NSSC-I memory (Step 2). QUEEN is activated by a burst of commands to the NSSC-I hardware which instruct the computer to halt itself, then restart using code from the high memory where QUEEN has been loaded. In Step 3, QUEEN loads Baseline 4.0 over 3.12E into low memory in small pieces that it receives from the ground, as it performs the telemetry collection, intercomputer communications, and safing protection functions described earlier. Finally, when 4.0 loading is complete, another burst of commands is sent to the NSSC-I hardware instructing it to halt, then restart - this time from the low memory to which the new software has been loaded. Under Baseline 4.0 control, stored command memory is reclaimed by loading a fresh supply of stored commands over the memory that QUEEN occupied, and normal operations are resumed (Step 4).

4. Applicability to Other Missions

The concept of a mini-executive for a spacecraft computer is not new to HST. The concept was used on the International Ultraviolet Explorer (IUE) and Solar Maximum Mission (SMM). IUE dedicated one 4K bank as a backup mini-executive to be used in case of problems with

the main 8K executive. Its functionality included telemetry collection and attitude control. It has been used to reload the main executive software. SMM used the concept during the repair mission, to allow software loading of the computer, thus reducing the time required to reload. SAMPEX, the first of the small explorer series of spacecraft, was launched in June of 1992. It includes a 386 processor with 512K of memory. SAMPEX flight code is written in the C language. SAMPEX has adopted a flight software modification philosophy which allows for complete tasks to be replaced. The compiled task is uplinked to a spare area of memory, and appropriate pointers in the assembly language image are updated. This philosophy avoids the timeline disruption of a full reload, while preserving the advantages of fully integrated code changes (except for the task pointers).

It appears that the QUEEN concept is reasonable to pursue for missions with on-board computer software requiring frequent and/or extensive changes where memory is constrained. HST is developing a similar mini-executive for the DF-224, planned to be used in a flight software installation in April of 1993. For future missions where flight software maintenance activity is expected to be high, provision of adequate spare memory would appear to allow for better solutions. Spare memory could be used for a full reload of the flight software image, without interruption to ongoing activities, or a task reload philosophy, like SAMPEX, could be adopted.

5. Conclusions

The issues involved in installing enhancements into the NSSC-I payload control computer of the Hubble Space Telescope have been examined, along with the advantages and disadvantages of the various methods available to achieve this end. QUEEN was an attempt to incorporate the positive aspects of these various methods into a single working product and procedure designed to aid science planning and spacecraft operations by providing nominal payload health and safety protection and telemetry, more efficient use of spacecraft resources, and of course, the desired NSSC-I software upgrades afforded in the new Baseline 4.0 software. QUEEN was successfully utilized to accomplish the reload of the HST payload computer in June 1992. It is planned to be used again for the NSSC-I Version 5.2, software that will support the first HST servicing mission (scheduled for installation in November 1993).

In a software development sense, QUEEN was a significant and meaningful activity. It embodied such state of the art concepts in software engineering as reusability (with its inherited software core) and total quality management (with the interactive spiral development process of software and procedures among the NSSC-I software engineers, science planners, and spacecraft operators). Yet these current trends were applied to a vintage hardware and assemble language software environment - ultimately yielding a useful and reusable utility.

REFERENCES

1. NASA, Goddard Space Flight Center. *Multimission Modular Spacecraft On-board Computer Flight Executive Technical Description*, S-700-56, July 1982.